NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

OPTIMIZING THE ROUTER CONFIGURATIONS WITHIN A NOMINAL AIR FORCE BASE

by

Marcus E. McNabb

September 2009

Thesis Co-Advisors: Ray Elliott
Craig Rasmussen

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Optimizing the Router Configurations within a Nominal Air Force Base		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Marcus E. McNabb			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** A	

13. ABSTRACT (maximum 200 words)

Routing information is a delicate balance between theoretical ideas such as optimization of flow and shortest path routing and the reality of threats to the security of the network. The way in which information is routed can be massaged to give the best solution, that being one that aptly satisfies both of these constraints. But are the routers on a nominal Air Force base set up with all of these things in mind? The basis for this study will be graph theory, which will supply the tools necessary to analyze the underlying mathematical construct of the router configuration. A nominal Air Force base will be constructed with all of the functions and organizations that one would find on a nominal Air Force base organized in a nominal physical fashion. The optimal router setup for such a base will then be developed. From there, this design's feasibility will be analyzed from cost, security, and implementation standpoints. The scope of this research will be the routing of all traffic internal to the base—i.e., traffic originating or terminating off base will not be examined.

14. SUBJECT TERMS Network Architecture; Graph Theory; Spanning Tree; Network Security			**15. NUMBER OF PAGES** 63
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

OPTIMIZING THE ROUTER CONFIGURATIONS WITHIN A NOMINAL AIR FORCE BASE

Marcus E. McNabb
Captain, United States Air Force
B.S., U.S. Air Force Academy, 2004

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN INFORMATION WARFARE SYSTEMS ENGINEERING
and
MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author: Marcus E. McNabb

Approved by: Ray Elliott
Thesis Co-Advisor

Craig Rasmussen
Thesis Co-Advisor

Carlos Borges
Chairman, Department of Applied Mathematics

Dan Boger
Chairman, Department of Informational Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Routing information is a delicate balance between theoretical ideas, such as optimization of flow, shortest path routing, and the reality of threats to the security of the network. The way information is routed can be massaged to give the best solution—that being one that aptly satisfies both of these constraints. But, are the routers on a nominal Air Force base set up with all of these things in mind? The basis for this study will be graph theory, which will supply the tools necessary to analyze the underlying mathematical construct of the router configuration. A nominal Air Force base will be constructed with all of the functions and organizations that one would find on a nominal Air Force base organized in a nominal physical fashion. The optimal router setup for such a base will then be developed. From there, this design's feasibility will be analyzed from cost, security, and implementation standpoints. The scope of this research will be the routing of all traffic internal to the base—i.e., traffic originating or terminating off base will not be examined.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I want to thank my wife, Kori, for all of her love and support throughout my two years at the Naval Postgraduate School. Without her, this would have been a far more painful stay. To the rest of my family, thank you for all of your support and wisdom.

I would also like to thank all of the professors at the Naval Postgraduate School who devoted their time and effort to teaching me the valuable skills and knowledge that I have acquired during my time here. In particular, I want to thank my advisors, Ray Elliott and Craig Rasmussen, for the extra time and energy that they devoted to helping me complete my thesis.

Finally, I would like to thank the Air Force Institute of Technology's Civilian Programs office and the Air Force Personnel Center for affording me the opportunity to attend the Naval Postgraduate School and pursue a dual degree.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM

Routing information is a delicate balance between theoretical ideas such as optimization of flow and shortest path routing and the reality of threats to the security of the network. The way information is routed can be massaged to give the best solution, that being one that aptly satisfies both of these constraints. But, are the routers on a nominal Air Force base set up with all of these things in mind? The basis for this study will be graph theory, which will supply the tools necessary to analyze the underlying mathematical construct of the router configuration. A nominal Air Force base will be constructed with all of the functions and organizations that one would find on a nominal Air Force base organized in a nominal physical fashion. The optimal router setup for such a base will then be developed. From there, this design's feasibility will be analyzed from cost, security, and implementation standpoints. The scope of this research will be the routing of all traffic internal to the base—i.e., traffic originating or terminating off-base will not be examined.

B. RESEARCH QUESTIONS

1. How is a nominal Air Force base set up, from both a physical and a network perspective?
2. What functions/organizations are on a nominal base?
3. What are the constraints to the problem (physical topology, network topology, etc.)?
4. Theoretically, how should the routers be placed and connected on such a base?
5. Is this a feasible design?
6. What, if any, security concerns are raised by such a design?

After these questions are answered, the author will search for a way to increase efficiency without sacrificing security.

II. BACKGROUND

A. DEFINITIONS

A *graph* G consists of a non-empty set V of *vertices*, which are points or nodes in the graph, and a set E of *edges,* where an edge is a connection between precisely two vertices. The *order* of G refers to the cardinality of V, and the *size* of G is the cardinality of E. For the purposes of this thesis, the vertices represent individual pieces of equipment (routers, computers, etc.). An edge is *incident* to its own endpoints, and those endpoints are said to be *adjacent* and are also called *neighbors*. Given neighbors x and y, the edge joining them is denoted as the edge xy. Note that an empty edge set means that the graph has no connectivity. In addition to being uninteresting, this setup is also unrealistic and contradictory to the purpose of this examination. The author is trying to enhance the connectivity on a nominal Air Force base, which is impossible with no connections.

As alluded to above, a *connected* graph is one in which there exists at least one path joining any two vertices in the graph. Note that this does not imply that there exists an edge between each vertex (such a graph is called *complete*), but that one may traverse a sequence of edges in some manner to reach the destination. If at least two vertices exist for which this cannot be done, then the graph is called *disconnected*. Furthermore, a graph is *k-edge-connected* if the removal of any set of edges with cardinality smaller than k leaves a connected graph. Given a connected graph G, if there is an edge e in G such that its removal results in G being disconnected, then e is a *bridge* in G. Given a vertex v or an edge e in G, the notation G - v denotes the graph containing all of the vertices and edges of G, except for the vertex v and its incident edges. Similarly, given an edge e in G, G - e denotes the graph containing all of the vertices of G and all of the edges of G except for e.

Another idea in graph theory utilized in this thesis is that of a *cut vertex,* which is any vertex whose removal (and thus the removal of its incident edges) results in a disconnected graph. A graph with at least one cut vertex is called *separable*, and a graph with none is called *non-separable*. A *cycle*, C, is a collection of at least three vertices and their incident edges such that one may choose any vertex on the cycle and reach the starting vertex by travelling along the edge set without repeating a vertex or edge. A cycle consisting of n vertices is denoted by C_n. A slightly relaxed version of a cycle is a *closed trail*, which has the same restrictions as a cycle except one is allowed to repeat vertices but still not edges.

Graphs may be *weighted*, meaning each edge is assigned a nonnegative number by a function, w, mapping the edge set of a graph to the nonnegative real numbers. The value assigned to an edge is generally associated with the relative cost to traverse that particular edge. The weight of an edge in a network graph is generally determined by physical characteristics of that edge, i.e., bandwidth, length, data rate, and loss. The weight may also affected by an administrative decision. For instance, a particular edge may be reserved for a particular function, so it is administratively assigned a high cost to discourage traffic from flowing on that edge. When travelling along paths with multiple edges, the path weight is merely the sum of the weights of each individual edge in the path. An optimization of the network modeled by such a graph would generally seek to minimize the total weight of the graph in some way.

This thesis deals heavily with a specific type of graph called a *tree*, which is a connected graph containing no cycles. Every tree can be drawn, as the tree in Figure 1 is drawn, that being one chooses a “starting point” for the tree and call it the *root vertex*, in this case, j. The root vertex generally can be any vertex, but for reasons that will be apparent later, this thesis will restrict the choice of the root vertex to vertices of degree at least 2. Given two adjacent vertices, the vertex nearest the root vertex is called a *parent* and the one farthest from the root

vertex is called the *child*. Two non-adjacent children with the same parent are called *siblings*. Given two vertices that can reach one another without traversing the root vertex, the one closest to the root vertex is called an *ancestor* of the other, which is called a *descendant* of the ancestor. The root vertex is, by definition, an ancestor to every vertex and every vertex is a descendant of the root vertex. Each vertex is also trivially both ancestor and descendant to itself. A *leaf* is any vertex of degree 1. Notice that leaves are defined such that they exist independent of a root vertex. A *branch* of a vertex v is a subtree rooted at a child of v. If v has no children, then it has no branches. If not specifically stated, then a branch refers to a branch of the root vertex, i.e., those vertices which are connected without traversing the root vertex. Note that every maximal path from the root vertex ends in a leaf. Trees have the important property that only one path connects any pair of vertices.

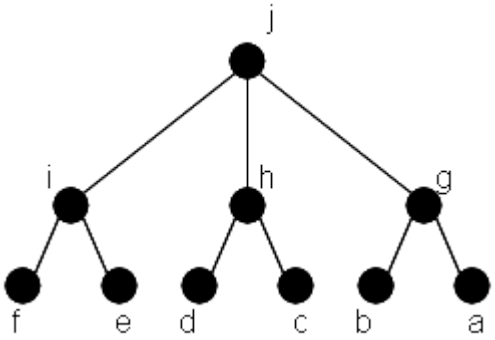

Figure 1. Tree example

This thesis will also discuss two additional specific types of trees: paths and stars. A *path* is a set of distinct vertices and the edges joining them, such that precisely two vertices have degree 1 (the starting and ending points) and every other vertex has degree 2. The path consisting of n vertices is called P_n. A *star* is a graph with a central vertex v, and the remaining vertices are adjacent to v and only v. Thus, v is adjacent to every vertex in a star. The star containing n vertices is called S_n.

Before delving into the subject matter, one more bit of graph theory is needed. Chartrand and Zhang [1] show that an edge e of a graph G is a bridge if, and only if, e lies on no cycle of G. This theorem can be relaxed a bit further to state that e is a bridge if, and only if, e lies on no closed trail of G. Proceeding as Chartrand and Zhang did with a proof by contradiction, let e be an edge connecting two vertices, say u and v, of a connected graph G. Since e is not a bridge, then G - e is also connected. Thus, there exists a path, P, from u to v in G - e. Then, P together with e forms a cycle (and thus a closed trail) in G. To verify the converse, suppose that e is an edge between u and v, which are again vertices of a connected graph G. Also suppose that e lies on a closed trail in G. Then again there is a path, P, from u to v in G that does not contain e. To show that G - e is connected, let x and y be any two vertices in G - e. Since G is connected, there is a path, Q, from x to y in G. If e is not on Q, then Q is a path from x to y in G - e as well. If e does lie on Q, then replacing e in Q by the path P produces a walk between x and y in G - e. Hence, there exists a path from x to y in G - e. It follows that G - e is connected and, therefore, e is not a bridge.

Lastly, a bit of networking terminology is necessary before proceeding. The Open System Interconnection Reference (OSI) model is a commonly used descriptor for a network. It has seven layers, but this thesis will only discuss layers 1, 2, and 3. Layer 1 is the physical layer and it corresponds to the network hardware. Layer 2 is the data link layer and it specifies how to organize and transmit data over the physical network. A switch is a commonly used device to transmit data across layer 2. Layer 3 is the network layer and it refers to how addresses are assigned and data packets are forwarded across the network. Routers are devices commonly used to communicate at layer 3. A collapsed core, or multi-layer, model is one in which a physical device is able to operate on layers 2 and 3 [2], [3].

B. NOMINAL AIR FORCE BASE

For this study, the author shall consider a nominal Air Force base to consist of a typical flying wing and all of the supporting functions necessary for such a wing. The active duty bases under Air Combat Command and Air Mobility Command constitute the majority of the Air Force's flying units. A survey of said bases revealed that the average base had a workforce, including active duty, reserve, and National Guard airmen as well as civilians, of approximately 8,500 [4]. Given a typical router, assume 20 switches attached to each router and 48 users per switch, which in turn dictates 9 routers.

On this nominal Air Force base, no geographical constraints exist. So the base's physical organization will contain two routers near the flightline to support the flying units, maintenance units, etc. Two additional routers are located a fair distance away to represent the ammunition unit and the medical unit. The remaining five units with their routers will be clustered in some central location as is typical. The following diagram will illustrate the router setup:

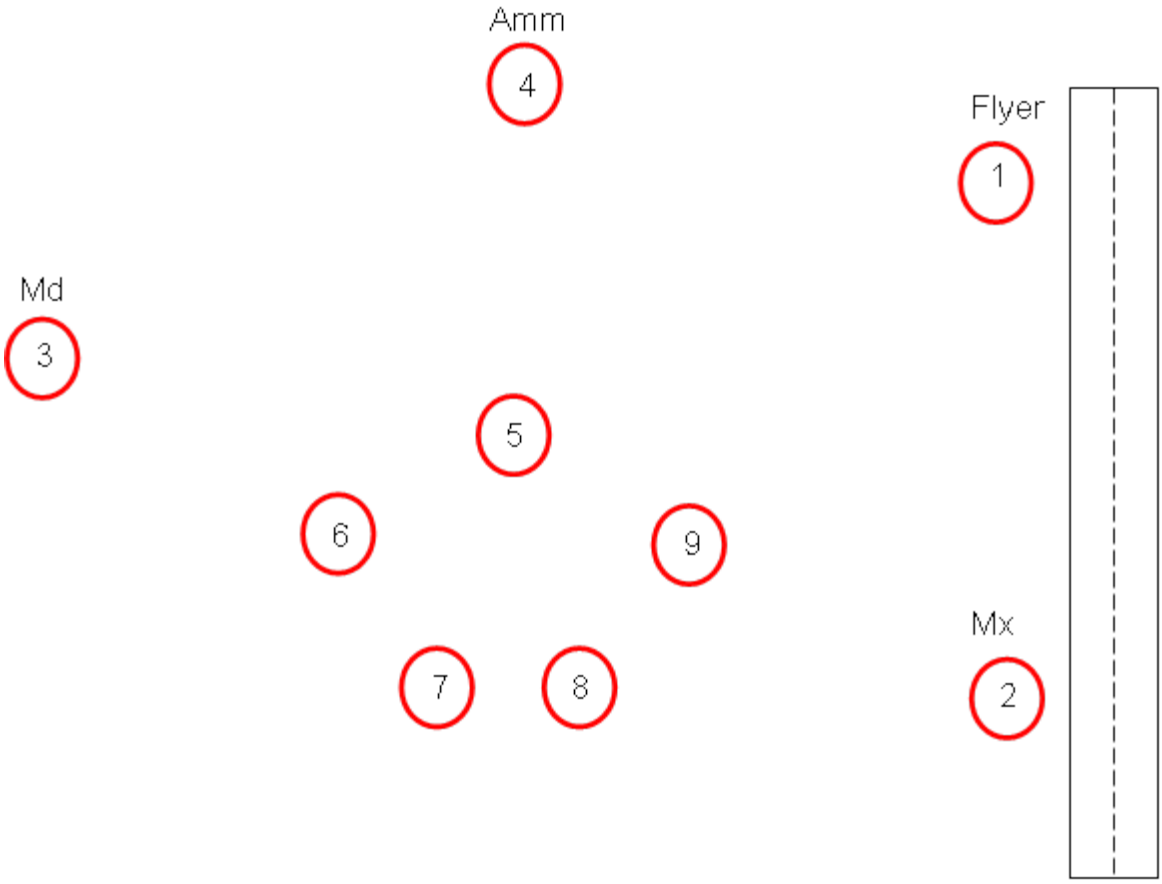

Figure 2. Router locations on a nominal Air Force base

Each location may be subdivided even further, but assume that this is the design at the top level of the network. Also assume the same hardware at all locations and the same bandwidth and data throughput rate on each line. Given these restrictions, it makes sense to make the weight of each path directly proportional to the length of said path. So a path that is twice as long as some other path should carry a weight that is twice as large. Notice that data throughput rate is constant for the lines regardless of length, so it does not provide a good discriminator by which to base edge weights. It will take the information longer to travel down a longer line, but the nominal Air Force base will only cover an area of several square miles. Since the information will be traveling at approximately the speed of light, the time difference of arrival

between two lines of differing length will be negligible. The two main factors influencing the weights are installation costs and losses. It is reasonable to assume that laying twice as much line will cost about twice as much in materials and labor, lending credence to the weighting scheme. This particular weighting scheme also accurately represents the power losses encountered over a transmission line since the relationship between loss and distance is linear when loss is measured in decibels. This is significant because an increase in power loss will result in an increase in the number of bit errors, which in turn could result in the information being incorrectly decoded on the receiving end. Thus, one concludes that this weighting scheme is a reasonable model.

This leads to an important observation: the weight of the edge ab is no more than the sum of the weights of edges ac and bc, where a, b, and c are adjacent vertices of a graph. This is the graph analogue of the triangle inequality in the plane $\mathbb{R}^2$ and merely states that the edge incident to two vertices is the smallest-weight path available between those two vertices [5].

Another important point to make here is that the work set forth in this thesis is relevant and valid for any weighting scheme. This particular scheme allows for exploration of a weighted graph, but the scheme itself is irrelevant. The results hold for any weighted graph regardless of how the edge weights were derived.

THIS PAGE INTENTIONALLY LEFT BLANK

III. NETWORK OPTIONS

A. CLASSICAL NETWORK ARCHITECTURES

A brief survey of the literature on network architecture yields four basic solutions to the problem: a mesh configuration, a star configuration, a ring configuration, and a partial-mesh configuration [2]. A mesh configuration pairwise joins each pair of vertices, while a partial-mesh configuration is merely some subgraph of the mesh configuration. The best possible solution to maximize connectivity amongst these routers is to connect each organization to every other giving a complete mesh configuration and a complete graph, but this option is quite expensive.

Given n vertices, the graph requires $\binom{n}{2}$ connections, which is on the order of n^2. To illustrate the expense, consider the connection between routers 1 and 3. Remembering that the path weights are directly proportional to path lengths, one can see that an edge joining routers 1 and 3 will be quite expensive. If both were connected to router 5, router 5 could act as an intermediary between the two and eliminate the expensive connection between routers 1 and 3. Therefore, many edges in a complete mesh configuration offer some measure of redundancy but sometimes at a very high cost. Thus, one may be able to reduce the cost while maintaining network connectivity. So using a mesh configuration leaves very little flexibility in the construction of the network. Perhaps a better solution is to maximize the use of routers and switches as an alternative to some of these connections.

The other three options, ring, star, and partial-mesh configurations, offer different perspectives on how to best achieve this balance between efficiency and productivity. A star configuration is precisely isomorphic to a star graph, S_n. This setup offers the advantage that some central router is directly connected to

every other router; this also means that any two routers are at most two hops apart. The downside is that the central router has a massive workload. It must handle every bit of traffic that travels between two routers. This threatens not only performance, but also security. It offers a single target by which an attacker may disable the entire network. Obviously, this is not a desirable trait for a network.

A ring network is a set up as a cycle containing each router. Thus, each router passes information to its neighbor, which in turn passes the information to its neighbor until the information reaches its desired destination. This setup offers a level of redundancy, in that it is 2-edge-connected and non-separable. If any connection or router is unavailable, information can just travel the opposite way along the cycle. The only downside to this configuration is its rigidity. Each router must have exactly two neighbors. This may not be advantageous in some situations, but overall this is a viable solution.

Finally, one arrives at the partial-mesh configuration. It is very flexible because it is not predetermined by any specific design. It is simply some collection of edges amongst the vertices. Thus, one may design a network as one sees fit, with all of the properties one desires. In the next section, the author will address exactly how this is done. After deciding exactly what a partial-mesh network will look like, the author will conclude his debate between the ring and partial-mesh configurations.

B. MATHEMATICAL SOLUTION

The next step is to address the requirement: a robust network. In other words, what is the minimum requirement to create a network such that any two routers are connected by some path? The requirement is that each router is able to communicate with every other router. Letting each router represent a vertex on a graph, one concludes that they must form a connected graph. The minimal such way is by a minimum spanning tree (MST) of the original complete graph.

A spanning tree is any tree with the same vertex set as the original graph. Such a tree with a minimum total edge weight is an MST. The size of any spanning tree is one less than the total number of vertices in the graph. Any graph of smaller size must be disconnected.

One of the main advantages to basing the research on an MST is its simplicity. Prim and Kruskal developed simple algorithms for finding an MST. To use Prim's algorithm, choose any vertex of a graph G and the lowest weigh edge incident to it. To select the remaining edges, choose the lowest weight edge that has shares exactly one endpoint with the edges already selected. When no longer possible, the edge set with its incident vertices forms an MST. Kruskal's algorithm operates in a similar fashion [1]. For example, consider the graph given in Figure 3, where the letters represent the vertices and the numbers represent edge weights.

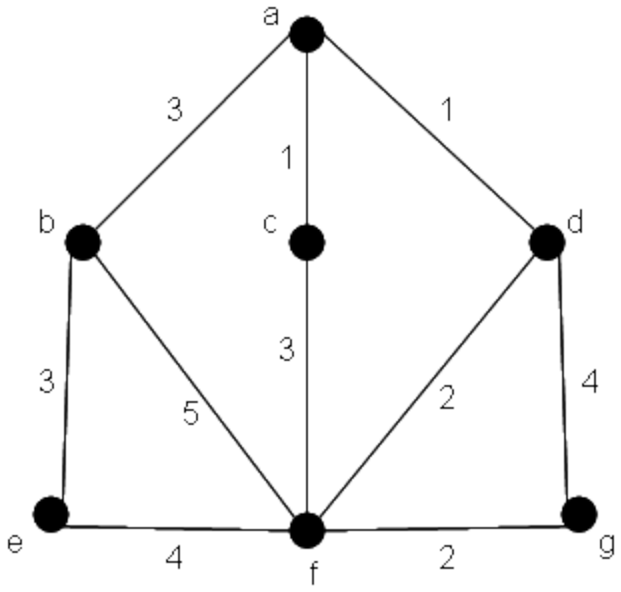

Figure 3. Example for Prim's Algorithm

Using Prim's algorithm, choose an arbitrary starting point, say vertex f. Both edges df and gf have weight 2, so arbitrarily choose one, say, edge gf. Now

one must choose the lowest weight edge with exactly one of g or f as an endpoint. In this case, that edge is edge df. Proceeding in this fashion, then choose the edges ad, ac, ab, and be, giving an MST as pictured in Figure 4 [1]. Note that there was a choice for the first edge, which turned out in this case to be a moot point because the edge not chosen was added to the MST, anyway. This may not always be the case, and as addressed later, a wise choice may yield the best possible MST.

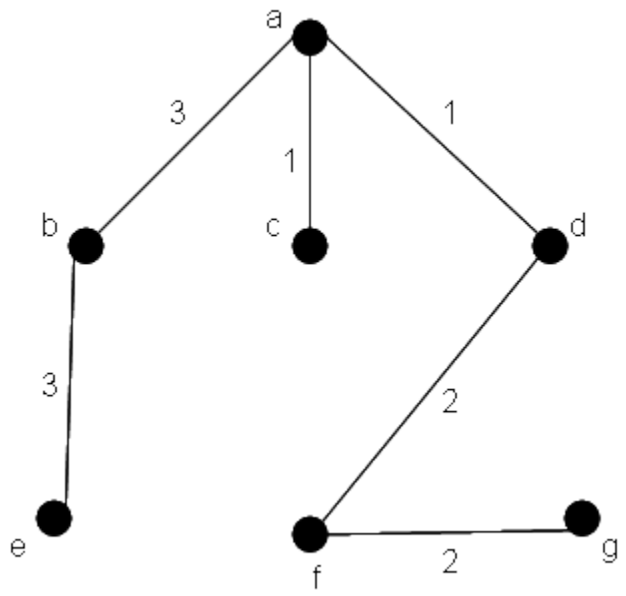

Figure 4. Result of Prim's Algorithm

A MST in the mathematical sense can also model some issues from a practical standpoint. The nominal Air Force base will contain two servers: one for e-mail applications and one to act as a gateway to the outside world. Each of these servers will be high traffic points on the network. Suppose it has a vertex v_1 which is four hops from the e-mail server v_2. Then, as a consequence of the fact that any vertex pair in a tree is connected by a unique path, all of the traffic between v_1 and v_2 will traverse the three hops between them. Then the routers connecting the two must handle not only their own traffic, but also all of the traffic between v_1 and v_2. Thus, the intermediate routers are in effect servicing a much

larger workload than v_1 and v_2, raising some question of the methodology. Alternatively, one could collocate these two servers and connect every router directly to each of these two servers. But this would almost surely come at a much greater cost than an MST, and it is not necessarily the case that every vertex must be a single hop away from these servers. Perhaps it would be best to just limit the distance to something like 2 or 3, thus giving a balance between the cheapness of a tree and the performance of a direct connection to these two servers. This configuration would give something very similar to a star configuration, placing a very heavy load on one or two central vertices and also limiting the choices in which edges to construct. But recall that in the star graph, the central router was forced to handle an extremely heavy workload, presenting both network and security issues. These conclusions lead the author to choose a different approach.

The next question is whether to let services drive the network design, or should the network design dictate the location of services? In other words, should it be decided first where to place the network and gateway, and then build the network around those locations? Or should an MST be built first and then choose the best locations on the MST for these servers? Since the location of these servers is undoubtedly driven by factors far beyond the scope of network design, assume that their locations are predetermined and the network must be designed such that it best meets the needs under those constraints. Also assume that the two servers are physically collocated, i.e., physically housed in the communications building.

Notice that the servers' physical proximity does not necessarily dictate that they be *virtually* near each other. So this creates another decision: should they be virtually collocated, adjacent, or as far apart as possible? Consider the ramifications of each situation. If they are virtually collocated, then that means a single router will handle all of the traffic through these two servers. This presents an immediate drawback, in the fact that this router will experience an enormous

workload. Alternatively, suppose it is near the “edge” of the network; this creates a situation where routers must act as intermediaries by handling traffic between other routers and the servers. If one locates the servers near the center, this situation improves. However, remember that the graph is only a tree, meaning that each pair of vertices is joined by a unique path. If two groups of routers are connected independently to the router handling the servers, then this router must also handle traffic between these two groups of routers. So the author begins to think that perhaps collocating the two may not be such a good idea.

So what about making them adjacent? This will not fix any of the problems previously discussed. A router (or group of connected routers) cannot be connected to both a router handling e-mail and a router handling external traffic, as this would create a loop. Thus, one could only connect to one of the routers, which would then again handle all of the traffic for both servers.

So while it may seem quite counterintuitive, think about locating the servers on opposite ends of the network. Consider the most extreme example, a network isomorphic to P_n, with two routers handling the servers represented by the endpoints. As before, the router adjacent to the endpoint that handles e-mail must send its data through every other vertex to reach the gateway router. Thus, the routers nearer the gateway handle more gateway traffic than just their own clients. But similarly, the routers near the e-mail router will handle more of the e-mail traffic. Thus, if these two routers deal in similar data quantities, the network has a natural load balance. Also, consider the typical data traffic on a network. For e-mail, the originator sends a message, comprised of x bytes, to the router. The e-mail router then forwards that message to the recipients. Thus, if the message has y recipients, a total of xy bytes are sent from the e-mail router to various clients. Similarly, an Internet traffic exchange typically consists of the client sending a small request packet and the server responding with a much larger amount of data. In each case, the majority of the traffic is flowing downstream from the server to the clients. This plays into the author’s hand

perfectly because the network is now operating in a duplex mode. Thus, the majority of the e-mail traffic is travelling one way while the gateway traffic is going the opposite way, naturally taking advantage of the network's duplex capabilities.

The conclusion here is that the routers must be physically collocated but virtually far apart. Thus, when building the MST, if given the option, one should stretch the network as much as possible to resemble P_n with the leaves representing these two servers.

C. VIABILITY OF MST

The most viable situation in which to use an MST configuration is one in which the network is actually governed by a layer-2 protocol. This is most often seen in the collapsed core model using multi-layer switches, meaning each router is physically embedded within a switch. This network topology offers the advantage of the speed of layer-2 switching with the functionality of a layer-3 router. In this network setup each router would represent a virtual local area network (VLAN). Each switch/router hybrid device acts as a switch for its own VLAN, but then has a virtual interface programmed that allows it to route traffic with an outside destination to a specific location [6].

The Rapid Spanning Tree Protocol (RSTP) is a popular choice to maintain this type of network topology. Loop-free routing is the major issue here, which RSTP addresses by forming an MST across the network and then blocking any links which are not part of the MST. Thus, any edges outside of the MST exist purely for redundancy's sake and offer no additional throughput under normal operating circumstances, i.e., times when no links are down [6], [7]. Thus, since an MST is the cheapest solution and the protocol will force the network structure into a tree design, an MST is the most logical answer.

Common Spanning Tree Protocol (CSTP) is a protocol that merely extends the ideas of the RSTP to a network amongst multiple VLANs. Again, it

would develop a single MST and apply it to the entire network, thereby lending itself very well to the methods laid out in this thesis [8].

However, even in other situations, an MST is a perfectly viable candidate, particularly those where cost and speed are severe constraints. The MST gives the cheapest connectivity solution, and thus gives the connections on which one should focus available resources. It also gives the smallest total path weight, so if the path weight corresponds to network speed (it does in this case since the path weight is governed by path distance), then it also gives the fastest overall network. The MST may not be the best solution in hostile environments where robustness and multiple layers of redundancy are needed. This solution must be augmented to achieve these characteristics, and a better solution may exist by instead building directly to these characteristics instead of trying to augment an insufficient graph to achieve them.

D. BUILDING THE MST

Returning to the nominal Air Force base, assume that the e-mail server is serviced by the router represented by vertex 7 while the gateway server is serviced by the router represented by vertex 8. Recall that the path weights will depend on distance alone. The actual distance here is irrelevant; one only needs to consider relative distances between vertices. Thus, assign a weight of one to each of the shortest edges, those being the edges between vertices 5 and 6, 5 and 9, 6 and 7, 7 and 8, and 8 and 9. From there, the proper values for every other edge are derived based upon this starting metric, which are given in Table 1. Using Prim's Algorithm, construct an MST of the graph, which is shown in Figure 5 (numbers next to the edges denote edge weights).

End Points		Weight
1	2	2
1	3	8
1	4	4
1	5	3
1	6	3.5
1	7	3.5
1	8	3
1	9	2.5
2	3	8
2	4	5
2	5	3
2	6	3.5
2	7	3.5
2	8	3
2	9	2.5
3	4	5
3	5	3.5
3	6	3
3	7	3.5
3	8	4
3	9	4
4	5	4
4	6	4.5
4	7	5
4	8	5
4	9	4.5
5	6	1
5	7	1.5
5	8	1.5
5	9	1
6	7	1
6	8	1.5
6	9	1.5
7	8	1
7	9	1.5
8	9	1

Table 1. Edge weights

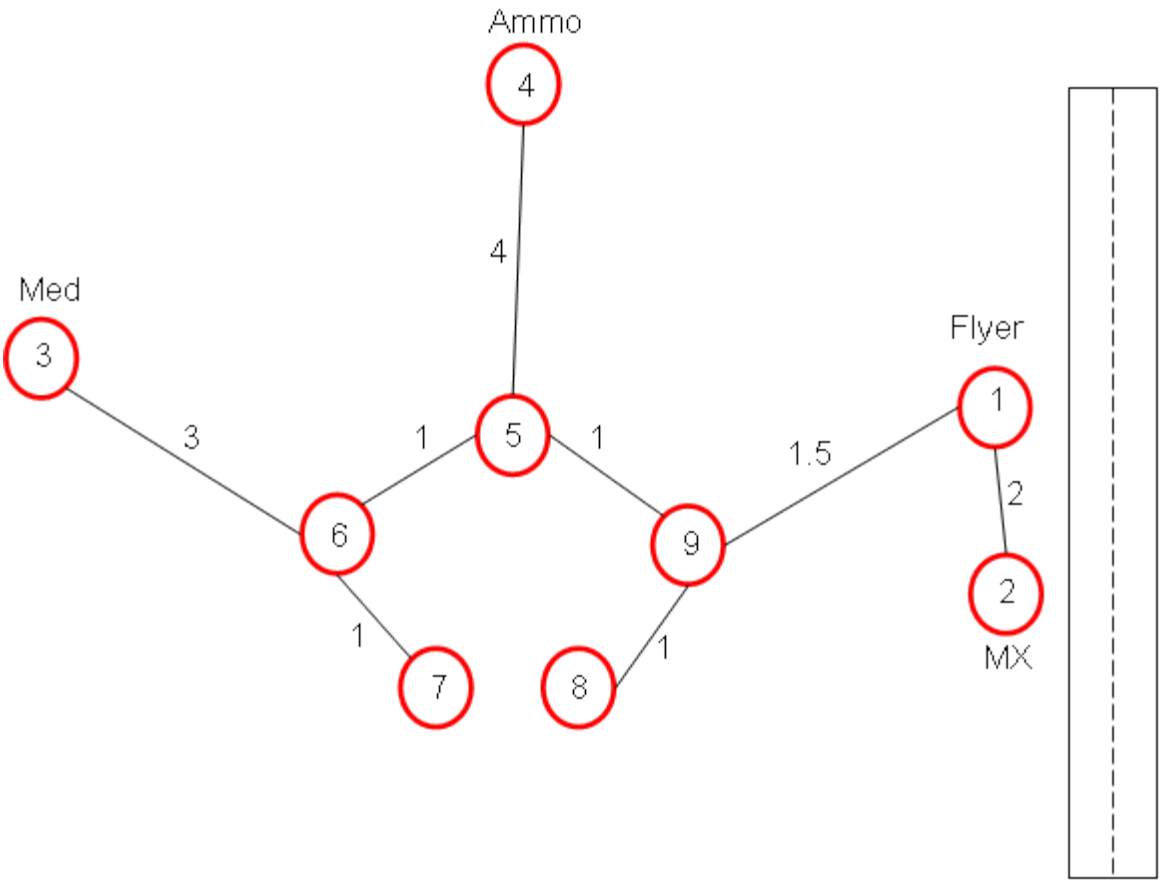

Figure 5. MST of the network of the nominal Air Force base

The total weight of the MST is 15.5. Notice that, according to the algorithm, single decision had to be made. Including each of the weight-1 edges would produce a cycle, thus one was excluded. According to the algorithm ties are broken arbitrarily, so the decision on which on to exclude is either based upon some human decision such as here, on a predetermined set of rules for breaking ties, or in a random fashion. Based upon the author's decision to virtually separate the e-mail server from the gateway server, and thus virtually separate vertices 7 and 8, the author chose to exclude the edge connecting vertices 7 and 8.

IV. ESTABLISHING REDUNDANCY

A. 2-EDGE-CONNECTED

The MST solves the connectivity problem, but the solution is not very robust, in the sense that a single failure of some connection between routers could produce network failure. For example, if the link between routers 6 and 7 fails, router 7 would lose its only connection to the remaining routers. Therefore, router 7 would be unable to communicate. This is especially bad because router 7 represents either the e-mail server or the Internet gateway. Thus, it is necessary to consider a way to construct a network such that this is not a concern by introducing some degree of redundancy. Since a connected graph does not accomplish this, the next logical candidate is a 2-edge-connected graph. By definition, a 2-edge-connected graph contains at least two edge-disjoint paths between any two vertices, so the removal of a single edge leaves at least one path between any two vertices.

This point in the discussion is a natural time to recall the candidacy of the ring topology as a solution to the requirements. Such a configuration would create a single cycle on which every router lies [2]. Recall that RSTP would then block the most expensive edge to guarantee a loop-free delivery, eliminating the ring and leaving a path. Also recall the argument of the efficiency of a single path connecting every router. A ring also gives the smallest 2-edge-connected graph over a set of vertices in terms of number of edges. However, this also puts some unnecessary constraints on the problem which may increase the overall cost in the long run. Consider a situation involving three neighbors a, b, and c in some larger graph G. Assume that b and c are each distance 1 from a and distance 2 from each other while all other vertices are sufficiently far away so as not to affect the decisions here. A ring topology would dictate that one connects two of these vertices to the remaining vertices of G and connect those two vertices to the vertex not connected. Thus, one could connect a and b to some

other vertices of G for a cost of, say, 4 each. Now one must join both a and b to c, raising the total cost to 11. But an MST configuration would give the freedom to connect only a to the remaining vertices of G for a cost of 4 again, then connect a to both b and c giving a total cost of only 5. Note that this configuration is not 2-edge-connected, and doing so would raise the cost above that of the ring. But remember that under normal operating circumstances only an MST will be active under the RSTP or CSTP. Thus, an MST actually produces a better daily situation, whereas the ring topology has better survivability traits. Operating under the idea that the network will most often be running under normal circumstances, pursuit of an augmented MST topology seems to be the most prudent choice.

Since deciding to stick with a MST, one must figure out how many additional edges are required to achieve 2-edge-connectivity. The author has shown that a complete graph is overly redundant and expensive, but an MST is not redundant at all. For reasons that will be apparent shortly, a tree can be made 2-edge-connected with the addition of precisely $\left\lceil \frac{k}{2} \right\rceil$ edges, where k is the number of leaves in the tree. This bound is also sharp, meaning that it cannot be done with any fewer edges.

Consider the following series of examples. First think of P_n. One may form a cycle (and thus a 2-edge-connected graph) by joining the two end-points of P_n, those being the two vertices of degree 1. Also, notice that these two vertices are the only leaves of P_n. Thus, $\left\lceil \frac{k}{2} \right\rceil = 1$ and the formula is validated for P_n.

Next, consider S_n, which has a central vertex v. Then the degree of v is $n-1$, meaning $n-1$ vertices are adjacent to v. Recall from the graph theory section earlier that if an edge is not a bridge, then its end points must lie on a common closed trail. The edges incident to v are exhausted, so one must join

other vertices. Joining two vertices in S_n-v puts those two vertices and v on a cycle of length 3. Thus, none of those edges is a bridge. Next, join two different vertices (other than the two already connected). Continue until all of the vertices are connected (or until one remains). Now consider v_1 and v_2 such that v_1 and v_2 are not adjacent. There must exist edge-distinct paths P_{1a} and P_{1b} from v_1 to v. Similarly, there exist edge-distinct paths P_{2a} and P_{2b} from v_2 to v. Joining these paths in pairs, i.e., P1a joined with P_{2a} and P_{1b} joined with P_{2b}, yields 2 edge-disjoint paths between v_1 and v_2. But both paths travel over v, thus v_1 and v_2 lie on a common closed trail. Therefore, any vertex to which an edge was added is now on a common closed trail with any other vertex to which an edge was added, so none of those edges is a bridge. The last step is to connect the single remaining vertex of degree 1, if such a vertex exists, by connecting it to any vertex. This puts it on a cycle with v and, by the same logic, on a closed trail with any other vertex. Thus, any two vertices lie on at least a common closed trail and there exist two edge-distinct paths between any two vertices. Therefore, the graph has no bridges. So how many edges were added? Since none were added to v, discount it. Also, a single edge accounted for two vertices, with a lone remaining vertex also requiring an additional edge. Notice that in this case, every vertex except v is a leaf. Thus, $\left\lceil \frac{k}{2} \right\rceil$ edges were added to S_n to make it 2-edge-connected.

One more observation given above must be generalized a bit further before the proof is presented. By the same logic used above, it easily follows that if any two vertices, say v_1 and v_2, lie on cycles that share at least one common vertex, then v_1 and v_2 lie on a common closed trail. This will prove useful in the following discussion.

The next consideration is a tree which does not conform to the star or path configuration. As before, connect the leaves in pairs (connecting the lone surviving leaf to any vertex if necessary). But some cases require restrictions and manipulations to the graph.

Given the MST in Figure 6, the formula predicts the addition of two edges for 2-edge-connectivity. The leaves are easily identified as vertices a, b, c, and d. If one was to connect these in no particular order, one could connect vertices c and d. This would force one to connect the siblings a and b, leaving one with a bridge joining e and f and forcing one to add a third edge. One avoids this situation by connecting a and b first to other vertices. Thus, choose a leaf with one or more sibling leaves as the starting point and connect it to a leaf which is not its sibling.

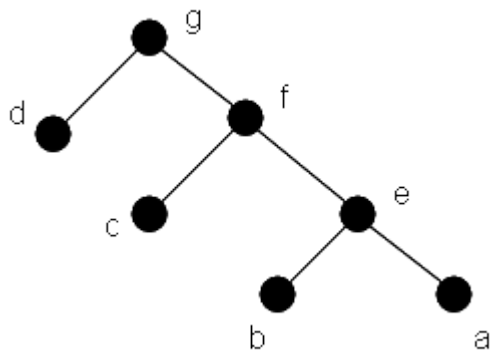

Figure 6. Illustration of sibling rule

Next, consider Figure 7. Following the previous convention of not joining siblings, one could join a and c, then b and d. But again this forces one to connect siblings e and f, leaving the edge ij as a bridge. Succinctly put, at each step, the chosen vertex should be a leaf with the maximum number of remaining sibling leaves. Then connect it to a leaf of a distinct root branch with the maximum number of remaining sibling leaves (again ties broken arbitrarily).

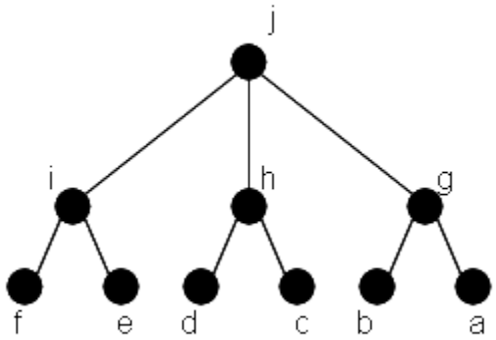

Figure 7. Illustration of necessity of the sibling rule

Finally, consider the situation given in Figure 8. The leaves are vertices a, b, c, d, and e, thus, according to the formula, requiring three additional edges to achieve 2-edge-connectivity. Notice that no leaf has sibling leaves. Following the restrictions already set forth, one may arbitrarily choose a pair of vertices to join. So choose d and e. Next, one again may choose any two remaining leaves to join, so choose a and b. Finally, join c to any vertex, say h. But this construction raises a problem. The edge ik is a bridge, thus requiring a fourth edge and violating the formula. To avoid this situation, do the following: if the only remaining leaves have no siblings, then choose a leaf from the branch with the most leaves remaining. Referring back to Figure 8, one could not begin as before by choosing vertices d and e because they are the lone leaves on their respective branches while leaves a, b, and c belong to the same branch. So one must choose either a, b, or c, and then join it with either d or e. Join c and d. Now only two branches remain, so one must join either a or b with e; join b and e. Notice that the remaining leaf is vertex a and the branch rooted at i on which a resides is isomorphic to P_n and a is not a child of i. In this case, join a with i to eliminate the final leaf. Otherwise, join a with any vertex.

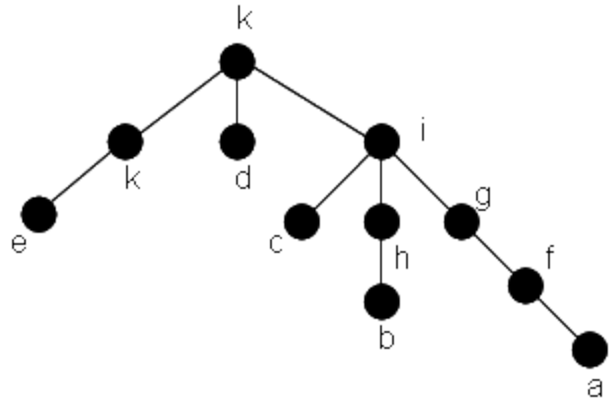

Figure 8. Illustration of necessity of maximum leaf rule

B. AN AUGMENTATION ALGORITHM

Let T be a tree with k leaves and root vertex r having degree at least 2. The author will now show that T can be transformed into a 2-edge-connected graph with the addition of precisely $\left\lceil \frac{k}{2} \right\rceil$ edges using the following construction:

1. Choose a leaf L with the greatest number of siblings that are also leaves. If no remaining leaves are siblings, then choose L such that the branch containing L has as many remaining leaves as possible. Let β be the branch of r on which L resides.

2. If there exists a leaf not on β, then choose such a leaf M with the maximum number of sibling leaves and join L with M. If no such leaves have siblings, then choose M such that the branch containing M has as many remaining leaves as possible and join L with M. Otherwise, let the first descendant of r with at least two branches replace r as the new root vertex and ignore the new root vertex's ancestors and their descendents (i.e., ignore all of the graph except the subtree rooted at the new root vertex). Proceed to step 3. If no such descendant exists, proceed to step 3.

3. If at least two leaves remain, repeat beginning at step 1. If a single leaf remains, join it to any vertex. If no leaves remain, the algorithm is complete.

The only restriction to the last step is that if a single remaining leaf is at the end of a branch isomorphic to a path, then join it with its closest ancestor that has at least two branches.

Refer to Figure 9 for the pseudocode for the algorithm.

```
Input: Tree T = (V,E) with root r such that the degree of r ≥ 2
K = {L ε V|deg L =1}
G = T
root = r
while K is not empty
        if no two branches of root contain leaves or no two branches of root exist
                let x be the child of r on the branch containing leaves
                if x in K
                        join x with any vertex to which it is not already joined
                else if |K| = 1
                        join x with the root
                else
                        root = x
                end if
        end if
        else
                choose a leaf x in K possessing as many siblings in K as possible
                        if no leaves possess siblings, choose x in K such that the
                        branch containing x has as many leaves in K as possible
                        end if
                let β be the branch of the root on which x resides
                choose a leaf y in K not residing on β possessing as many siblings
                in K as possible
                        if no leaves possess siblings, choose y in K not residing on β
                        such that the branch containing y has as many leaves as
                        possible
                        end if
                Join x with y
                K = K – {x,y}
        end if
end while
```

Figure 9. Algorithm Pseudocode

Proof: Let T' be an MST T of some graph that has been augmented by the above method. Let r be the designated root vertex of T. Suppose there exists a bridge b in T' that resides on some branch β of r. Then there exists at least one leaf L from T on β such that b belongs to the unique path joining L with r. Notice that β is not isomorphic to P_n for if it was then it would have been joined L with r, forming a cycle, consisting of the union of β with e, that contains each edge in β. Therefore, β has at least two leaves in T. Similarly, b is not incident to L because every path in T containing L must contain the only edge incident to L. Suppose that L was not a single remaining leaf in the algorithm. Then from the algorithm it must be the case that an edge e incident to L and some other leaf M was added.

Case 1: The first way the algorithm could have added an edge incident to L is if at some point L had the maximum number of sibling leaves remaining and L had at least one sibling leaf. Otherwise, refer to case 2.

Case 1a: The algorithm joined two leaves on distinct branches of the root. Then a cycle is formed by the union of the edge e with the path in T joining L with M, which contains the path joining r with L. Therefore, b is not in the path joining L with r, which contradicts the previous statement that it is.

Case 1b: The algorithm joined two leaves on the same branch of the root. So at this point in the algorithm, β contained at least two leaves, while every other branch of the root contained none. The only case where this may happen is if the size of the set of siblings of L is greater than the size of the leaf set not containing L and its siblings. If this is not the case, then if the algorithm annihilated the leaves of the other branches, then at some point they each had a single leaf remaining and the algorithm stipulates that in the case that no leaves have sibling leaves a leaf from the branch with the most leaves will be chosen. So suppose that it is the case that the size of the set of siblings of L is greater than the size of the leaf set

not containing L and its siblings. Since there was at least one leaf on a branch of r distinct from β (recall that the degree of r is at least 2), then at least one of the siblings of L was joined with some leaf on a distinct branch of the root. Therefore, no edge in that siblings path to the root can be a bridge. But since it is a sibling to L, they share a path to the root everywhere except at the edge incident to them. Also, remember that b is not incident to L. So this case could not occur.

Case 2: The next way a leaf is chosen is, if no leaves have sibling leaves remaining, then a leaf from the branch with the most remaining leaves is joined with a distinct branch of r containing the next most remaining leaves.

Case 2a: The algorithm joined two leaves on distinct branches of the root. Then a cycle is formed by the union of the edge e with the path in T joining L with M, which contains the path joining r with L. Therefore, b is not in the path joining L with r, which contradicts the previous statement that it is.

Case 2b: The algorithm joined two leaves on the same branch of the root. So at this point in the algorithm, β contained at least two leaves, while every other branch of the root contained none. Again, this could not be because if the algorithm annihilated the leaves of the other branches, then at some point they each had a single leaf remaining and the algorithm stipulates that in the case that no leaves have sibling leaves a leaf from the branch with the most leaves will be chosen. So this case could not occur.

If L was a single remaining leaf, then the graph was a 2-edge-connected graph with a pendant vertex. If L was at the end of a branch of some vertex that was isomorphic to Pn, then L was joined with its nearest ancestor that has at

least two branches. Then a cycle exists consisting of e joined with that branch, meaning no edges on that branch are bridges. Hence, the graph has no bridges. If L was not at the end of a branch of some vertex that was isomorphic to Pn, then the algorithm joined L with any vertex and similarly the graph has no bridges.

Next, the sharpness of this bound is shown. Consider a leaf L in T which is incident to an edge e. Since L is a leaf, then e is the only edge incident to L. If e is no longer a bridge, then it must be part of a cycle. The only way for this to occur is for one to join L with some other vertex. This is true for every leaf of T. Thus, one must have joined an edge to every leaf. Since each edge joins only two vertices, then it is the best case that one has joined two leaves with a single edge. But if one is left with a single remaining leaf after joining pairs of leaves, then it is still true that one must join this leaf to some other vertex or else the edge incident to it will remain a bridge. Thus, one cannot possibly construct a 2-edge-connected graph by augmenting edges to a tree with any fewer than $\left\lceil \frac{k}{2} \right\rceil$ additional edges. Therefore, this algorithm will produce a 2-edge-connected augmentation to any MST using the smallest number of edges. *Q.E.D.*

C. WEIGHTED GRAPH

The environment in which the work presented in this thesis resides deals with weighted graphs. Thus, the natural progression of the author's research should lead one to ask if the preceding formula can be molded into a weighted version. The immediate answer seems to be to implement the algorithm "as is" in a "greedy" manner—i.e., proceed as directed choosing the smallest weight edge available at each step. The algorithm allows for choices in the algorithm only in the case that multiple leaves have the same number of siblings. Given the limited nature of the choice along with the small size of the vertex set of the

graph in this thesis, it is most prudent to construct all of the possible augmentations and then compare the total weights.

Proceeding in this fashion with the nominal Air Force base, one finds that one is allowed to construct 32 augmentations to the MST, each of which consists of just three edges. However, the weights of these augmentations range from 8.5 to 14. Obviously, one would choose the smallest, which consisted of the edges between vertices 2 and 9, 3 and 4, and 7 and 8 with those edges having weights 2.5, 5, and 1, respectively. Recalling that the MST had a weight of 15.5, one now has a 2-edge-connected graph with 9 vertices, 11 edges, and a total graph weight of 24, depicted by Figure 10.

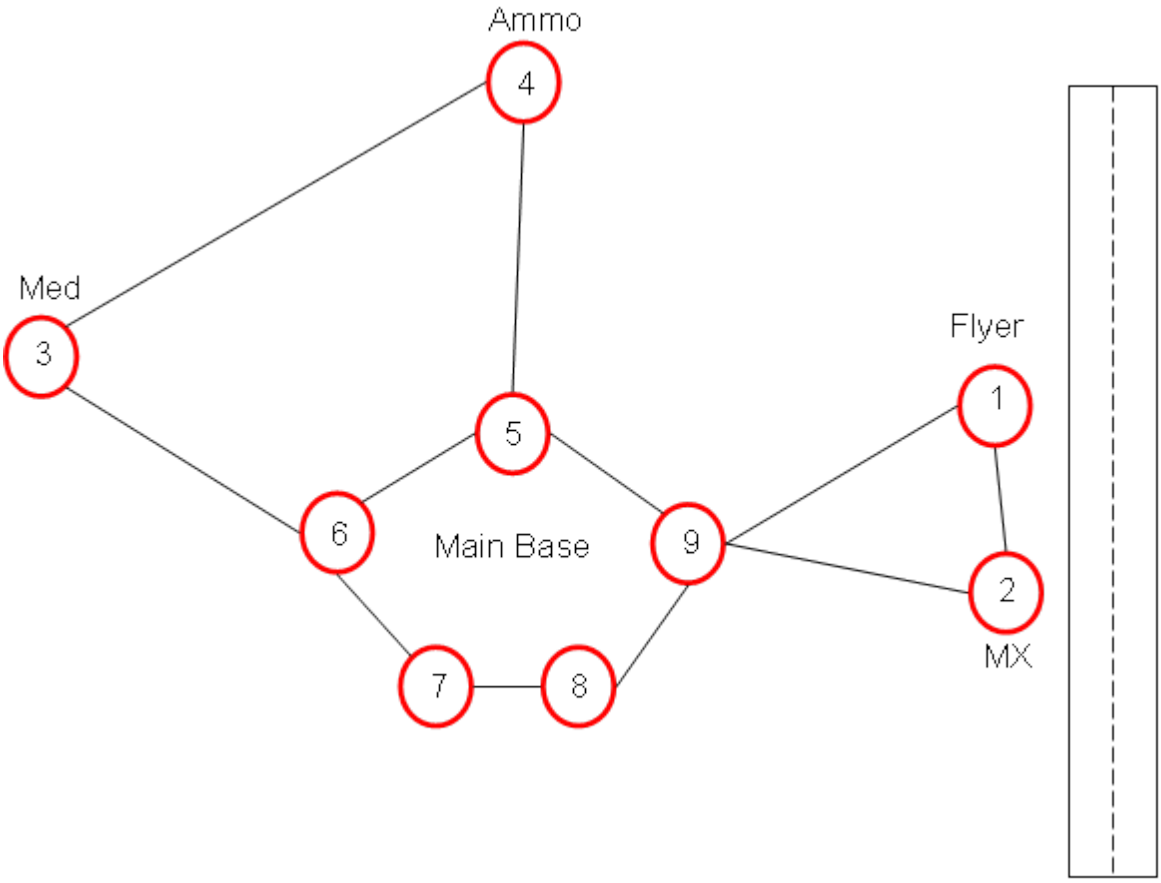

Figure 10. Two-edge-connected network on the nominal Air Force base

In this case, the smallest weight augmentation was unique. This may not always be the case and if not, the decision of which links to construct is mathematically irrelevant. Instead, more practical matters such as the organizations that will be serviced by a particular router will probably drive the decision.

THIS PAGE INTENTIONALLY LEFT BLANK

V. ANALYSIS

A. SPEED OF THE ALGORITHM

The problem of constructing an augmentation to some graph such that the result is k-connected is well-known to be an NP-complete problem. Even the more limited problem of finding only a 2-edge-connected subgraph of some larger graph is NP-complete [9]. But in this case, the problem was limited to augmenting a tree to reach 2-edge-connectivity by performing some operations on its leaves. How many operations are required?

Consider how the algorithm operates. At each step, it must make two calculations for each leaf: its parent and its ancestor nearest the root vertex. Thus, letting L be the number of leaves in the MST, the algorithm must make 2L calculations. Next, it must repeat, but it has two fewer leaves since the first step joined two leaves. Thus, it must now make $2(L-2)$ calculations, then $2(L-4)$ calculations, and so forth. In total, it repeats this step $L/2$ (or at worst $\frac{(L+1)}{2}$) times. Thus, in total, the result is $\frac{L+1}{2}(2L) = L(L+1) = L^2 + L < V^2 + V = O(V^2)$, where V is the total number of vertices in the graph. Given the fact that Prim's Algorithm is $O(E \log V) < O(V^2)$, then the total construction of the MST with the additional edges is $O(V^2)$.

B. ROOT VERTEX ANALYSIS

Throughout the development of the algorithm here, one recurring question arises: what is the impact of the choice of the root vertex? The heuristic answer would seem to be that one should choose a root to be a centrally-located vertex relative to the leaves of the graph, perhaps allowing one to keep the weights of the additional edges as low as possible. In Table 2, the numbers in this case

seem to back up this assertion. Recall from the graph that vertices 5, 6, and 9 are centrally-located relative to the other vertices while vertex 1 is not. Further research would be needed to make any sort of mathematical justification to the claim. In this case, the vertex set is small enough that the problem yields to the exhaustive method where all possibilities are calculated and the best is chosen.

Root	Average Total Weight
1	28.1
5	27.0
6	27.1
9	27.3

Table 2. Weighted MSTs of each available root vertex

VI. CONCLUSIONS

A. SUMMARY

The method for constructing a network architecture set forth here is simple; it is easy to design and implement. It is also cost-efficient and offers a single layer of redundancy with respect to the links. However, it is not very flexible and may not be adequate to support the traffic load on a fully operational base. This design is probably best suited for use in a tactical location because of its simplicity, cost-efficiency, and redundancy.

In short, this thesis used graph theory to analyze the options for a network on a nominal Air Force base and concluded that the best approach is to start with an MST (Minimum Spanning Tree) and augment it with some additional edges to make it 2-edge-connected. The MST is constructed using either Prim's algorithm or Kruskal's algorithm. Let the MST have root vertex r and k leaves. One may transform T into a 2-edge-connected graph with the addition of precisely $\left\lceil \frac{k}{2} \right\rceil$ edges by performing the following construction:

1. Choose a leaf L with the greatest number of siblings that are also leaves. If no remaining leaves are siblings, then choose L such that the branch containing L has as many remaining leaves as possible. Let β be the branch of r on which L resides.

2. If there exists a leaf not on β, then choose such a leaf M with the maximum number of sibling leaves and join L with M. If no such leaves have siblings, then choose M such that the branch containing M has as many remaining leaves as possible and join L with M. Otherwise, let the first descendant of r with at least two branches replace r as the new root vertex and ignore the new root vertex's ancestors and their descendents (i.e., ignore all of

the graph except the subtree rooted at the new root vertex). Proceed to step 3. If no such descendant exists, proceed to step 3.

3. If at least two leaves remain, repeat beginning at step 1. If a single leaf remains, join it to any vertex. If no leaves remain, the algorithm is complete.

The only restriction to the last step is that if a single remaining leaf is at the end of a branch isomorphic to a path, then it is joined with its nearest ancestor that has at least two branches.

This method yields a network design with at least one level of edge redundancy for all edges at a cost of $\left((n-1)+\left\lceil \frac{x}{2} \right\rceil\right)$ edges where n is number of vertices in the graph and x is the number of leaves in the MST.

B. FUTURE RESEARCH QUESTIONS

1. Non-separable Graphs

The construction undertaken in the previous sections is only practical under the assumption that the vertices are well protected. In other words, the vertices of the graph represent routers that sit in buildings with impenetrable physical and software security. If this is not the case, then one must consider the vertices a liability as well and take steps to introduce a level of redundancy to paths with respect to their vertex set. Just as the author moved from a connected graph to a 2-edge-connected graph in the case of vulnerable edges, in this case one should move from a separable to a non-separable graph. Thus, removal of any single vertex will not produce a disconnected graph. Frederickson and JaJa showed (non-constructively) that a 2-edge-connected graph can be transformed into a non-separable graph with no additional cost [5]. The advantages of such a design are immediate. If a graph is non-separable, then there exist at least two paths between any two distinct vertices in the graph such that these two paths share no common vertices except the first and last. An

important observation is that two such paths also do not share a common edge. Thus, if one constructs a non-separable graph, then one has also constructed a 2-edge-connected graph. Therefore, one can improve the robustness of the network at no additional cost. The disadvantages are less obvious but perhaps over-riding in importance. Moving to a non-separable graph will destroy the original MST. Recall that the foundation of the argument here is that an MST is the best starting point because the routing protocols will devolve any network into an MST. Thus, if one destroys the MST, then the protocols will devolve the non-separable graph into the best available MST of the non-separable graph, which will not necessarily (and indeed probably will not) match the MST of the network itself. Thus, the performance of the non-separable graph under optimal conditions, i.e., no vertex or edge failures, will fall short of the performance characteristics of the 2-edge-connected graph.

The next option is to further augment the 2-edge-connected graph into a non-separable graph. This, of course, will come with an increased cost in the form of additional edges. Such an addition would be made after weighing the pros and cons of each situation and the overall vulnerability of the network, specifically that of the vertices, and deciding that the network is sufficiently vulnerable to necessitate such additional resources.

Since the 2-edge-connected problem is NP-complete and is a sub-problem of the non-separable problem, then the non-separable problem is at least NP-hard. In fact, Frank first showed it to be NP-complete [9], and Kerivin and Mahjoub later showed that it is actually strongly NP-hard [10].

2. Edge Weight Impacts

These algorithms give the minimum number of edges, but what about minimum total cost? The author used an exhaustive method. The overall problem is well known to be NP-complete, but perhaps some specific scenario, such as the one presented in this thesis, would yield to an easier solution.

Perhaps a limit on the size of the network to some small number of vertices would sufficiently reduce the complexity of the problem so that it lends itself to a solution.

One could also consider the case that it may be advantageous to add two shorter edges instead of a single long edge (i.e., the analogue of the triangle inequality in $\mathbb{R}^2$ does not hold), but the complexity of this problem increases enormously. In such a small graph, the author was able to perform an exhaustive search to locate the lowest weight augmentation for his tree, but obviously this situation will quickly become unrealistic as the number of vertices increases.

3. The Multiple Spanning Tree Protocol (MSTP)

The author found that the best solution to his problem was to augment a single MST to give him a layer of redundancy. However, this is certainly a minimalist approach giving him a single tree carrying all of the network traffic. In a CISCO architecture, this is the CSTP. At the other end of the spectrum is the RSTP for a routing network. In this case, each router would build its own MST, giving the network administrators great flexibility in choosing over which paths routers should communicate; but this also comes at a much higher cost as the network must maintain as many trees as it has routers. The intermediate solution is MSTP in which the network designers may design around as many or as few trees as they would like, with each tree having any number of associated routers [8].

The simplest implementation of MSTP is merely the CSTP while the hardest implementation is RSTP. A realistic implementation will be a hybrid of these two protocols using some small number (greater than one) of trees. Each router would be able to receive from any edge but would only be able to transmit on an edge of its associated tree [8].

On the nominal Air Force base, one could use Prim's Algorithm to develop an MST, call it MST1, as was done earlier. One could then throw away the edges used in MST1 and, using Prim's algorithm again, develop a new MST, call it MST2, from the remaining edges. Since the available edges represent a complete graph, this development is nearly always possible if the graph contains at least four vertices. The only case in which this does not work is when MST1 is a star configuration because the central vertex has no available edges for MST2. Even in this case, the method will work by just developing MST2 amongst the vertices not including the central vertex. Consider removing the central vertex. The remaining vertices are still connected using MST2. Removal of any non-central vertex yields at least a subtree of MST1, and therefore, a connected graph.

The network would then be comprised of the union of these two trees. One could attach the various routers to whichever tree one would like; for example, one could attach the email router to one tree, the gateway router to the other tree, and then load balance the remaining routers onto the two trees. Another option would be to attach both the email and gateway routers to MST1 and the remaining routers to MST2. This application would only make sense if MST1 was physically capable of a larger data rate than MST2 since in this case MST1 would be handling the brunt of the workload for the network.

The union of two edge-disjoint spanning trees is 2-edge-connected, thus it meets the nominal Air Force base's redundancy requirements. If an edge was lost, then any path in that particular tree which was dependent upon that edge would be replaced by the path with identical endpoints from the other tree. Also notice that the union of n edge-disjoint spanning trees is n-edge-connected. Recall that the definition of a tree is that it contains a unique path between any two vertices in the tree. Thus, each tree contains one path between two particular vertices, giving the union n paths between those two vertices. Therefore, the union of n edge-distinct trees is by definition n-edge-connected.

Thus, this method will extend as far as one would like to take it until the edges are exhausted. However one also sees that this union of trees may be separable. Recall that each vertex in a tree is a cut vertex and it is possible that the union of n trees may also contain at least one cut vertex. Also, further research is necessary to determine whether the union of multiple instances of spanning trees is the cheapest implementation of MSTP. Some augmentation may accomplish the same goals using fewer edges and/or at a cheaper total cost.

Another option under the MSTP presents itself in the case of an isolated and relatively lightly used router, such as router four on the nominal Air Force base. In the judgment of the network designers, such a router may not need to be appended to two distinct trees. In this case, one could allow MST1 and MST2 to share a single edge connecting router four to the remainder of the network. This offers no redundancy for that particular vertex, for if that single edge is lost then router four is no longer connected to the network. This is an undesirable trait but perhaps worth the savings in cost. It merely adds an additional layer of flexibility in developing a network to meet the particular needs of a base.

The case of MSTP does not lend itself easily to a simple solution because the possibilities are wide-ranging and nearly limitless. Individual vertices and edges can be given a specific function within the network. If a network is designed using MSTP, the particular needs and uses of that specific network must be considered before deciding how to build a network. For example, say one tried to use a method similar to the author's to build an augmented MST. Since MSTP does not use a single spanning tree to relay traffic, then perhaps the augmented MST is not the correct approach. One may identify two or three critical vertices and build a network around those critical vertices in some method as to best protect those vertices.

4. Cloud Computing and Service-oriented Architecture

Another area for future study would be to analyze the impact of cloud computing, and the subsequent increase in servers, on the model. An immediate concern is that the author handled the weighted portion of the graph by a method of exhaustion. Of course, this method will become obsolete if the vertex set becomes too large. But, perhaps the more important question here is whether the method will transport to a cloud computing model. Does the cloud computing model offer enough versatility to render an augmented MST obsolete, and, even if so, will any of the basic principles carry over? Another topic for investigation along the same line of thought is the impact of a service-oriented architecture on the network design.

5. Simulation

The method proposed here is based on mathematical concepts but has not been applied to any sort of real-world scenario. A thorough test or simulation of the method is needed to validate the author's findings.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] G. Chartrand and P. Zhang, *Introduction to Graph Theory*. New York: McGraw Hill, 2005.

[2] U.S. Air Force Base Locator, http://www.airforce.com/contact-us/base-locator/, accessed June 1, 2009.

[3] G. N. Frederickson and J. JaJa, “On the relationship between the biconnectivity augmentation and traveling salesman problems,” *Theoretical Computer Science*, issue 19, pp. 189–201, 1982.

[4] D. Comer, *Computer Networks and Internets with Internet Applications (Ed. 4)*. Pearson Prentice Hall: Upper Saddle River, NJ, 2004.

[5] G. Donahue, *Network Warrior*. Beijing: O’Reilly, 2007.

[6] *Understanding Rapid Spanning Tree Protocol (802.1W)*. CISCO Systems, 2008.

[7] *IEEE Standard 802.1D*, IEEE, New York, 2004.

[8] *Understanding Multiple Spanning Tree Protocol (802.1S)*, CISCO Systems, 2007.

[9] A. Frank, “Augmenting graphs to meet edge-connectivity requirements,” *SIAM Journal for Discrete Mathematics*, vol. 5, no. 1, pp. 25–53, February 1992.

[10] H. Kerivin and A. R. Mahjoub, “Design of survivable networks: A survey,” *Networks*, vol. 46, no. 1, pp. 1–67, 2005.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. CDR Michael Herrera
   Naval Postgraduate School
   Monterey, California

4. Ray Elliott
   Naval Postgraduate School
   Monterey, California

5. Craig Rasmussen
   Naval Postgraduate School
   Monterey, California

6. Carlos Borges
   Naval Postgraduate School
   Monterey, California

7. Dan C. Boger
   Naval Postgraduate School
   Monterey, California

8. NETWARCOM/N1/N17
   Norfolk, Virginia